Database Security Using MARS Symmetric-key Encryption Algorithm

Swe Zin Moe, Win Aye
University of Computer Studies (Mandalay)
szmoemoe@gmail.com

Abstract

The need to protect database, would be an every growing one especially so in this age of e-commerce. Many conventional database security systems are bugged with holes that can be used by attackers to penetrate the database. No matter what degree of security is put in place, sensitive data in database are still vulnerable to attack. To avoid the risk posed by this threat, database encryption has been recommended. However encrypting all of database item will greatly degrade the performance of the database system. As an optimal solution, this paper presents a database encryption scheme that provides maximum security, whilst limiting the added time cost of encryption and decryption.

Key words: *Database, Database Security, Cryptography, Cryptographic Keys Encryption, Decryption, Access Control*

1. Introduction

In today's economy databases symbolize one of the most valuable assets. They form the basis for e-business, e-commerce, Enterprise Resource Planning (ERP) and other sensitive activities. Many organizations cannot work properly if their database is down; they are normally referred to as mission-critical system. Along with the wide application of databases comes the need for its protection. Universally, huge amount of effort, time and resources are been spent in trying to make database systems meet security requirements. These security requirements include:

- Prevention of unauthorized disclosure of information
- Prevention of unauthorized modification of I information
- Prevent denial of service
- Prevent system penetration by unauthorized person
- Prevent the abuse of special privileges

Designing a database that will achieve these security requirements is very difficult, since a database system processes large amount of data in complex ways. The result is that most conventional database systems have leaks that attacker can use to penetrate the database. No matter what degree of security is put in place, sensitive data in database are still vulnerable to attack. A remedy therefore is to turn to cryptographic means of storing data. Encrypting data stored in a database can prevent their disclosure to attackers even if they manage to circumvent the access control mechanism. Thus cryptographic technique can ensure excellent security for databases, by reducing the whole security process down to the protection of only few cryptographic keys. Nonetheless the time cost involved in encrypting and decrypting data items can greatly degrade the performance of the database system. A compromise solution must be found between performance and security, by encrypting only sensitive data in tables or column in the database. The objective of this paper is to propose a secure database encryptions scheme that provides maximum security, whilst limiting the added time cost for encryption and decryption. The encryption technique considered is MARS, which stands for multiplication, addition, rotation and substitution - Symmetric-key Encryption Algorithm, but the scheme is also applicable to other cryptographic techniques and standards.

2. Related Work

[1, 2] has contributed immensely to efforts towards providing Database Security through Cryptographic means. In [1] he provides solutions to the security problems of field based protection. In his paper he presents a comparative study on implementing encryption at various database levels i.e. table, attribute and field (element) levels. In [2] he tries to solve database integrity problem through cryptographic checksum.

[3] Proposed a system model using two trusted modules (security kernels), and use tags to ensure integrity. [4] Proposed a blend record and field techniques, based on remainder theorem. All the above approaches are different from this study in terms of structure, key management and the implementation procedures.

The rest of this paper is organized as follows. Planning a database encryption strategies are discussed in Section 3. Section 4 explains the

MARS Encryption Algorithm in detail. Implementation of the system is presented in Section 5. Section 6 provides some concluding remarks.

3. Planning a Database Encryption Strategy

To effectively secure the databases using encryption, three issues are of primary importance: where to perform the encryption, where to store encryption keys and who has access to encryption keys. The process of encryption can be performed either

1. Within the database, if the DBMS supports the encryption features, or

2. Outside the DBMS, where encryption processing and key storage is offloaded to centralized Encryption Servers.

3.1 Implementing encryption inside the DBMS

If encryption features are available within the DBMS product, the system can encrypt and decrypt data within the database and the process will be transparent to the applications. The data is encrypted as soon as it is stored in the database. Any data that enters or leaves the database, though, will be transported as clear text. This is one of the simplest database encryption strategies, but it presents performance trade-offs and security considerations that must be evaluated.

Encryption generally is implemented within the database through a "database procedure call" (the terminology varies by vendor). Some vendors support limited encryption capabilities through database add-ons. Other vendors may only provide all-or-nothing support for encryption either the entire database is encrypted, or nothing is. While this may make sense for protecting the backup copies, encryption of the entire database means additional processing is expended on non-sensitive data — an overkill situation resulting in unnecessary performance degradation.

3.2 Implementing Encryption outside the Database

If the potential for data exposure in the database or in transit between client and server concerns the system, a more secure solution is moving the encryption to the applications that generate the data. When the system use client/server application security protocols like SSL, sensitive data is in clear text form for the shortest possible time. Encryption is performed within the application that introduces the data into the system; it travels encrypted and can be stored encrypted at its final destination.

This approach can provide good end-to-end data protection, but may require changes to the applications to add or modify encryption and decryption capabilities. One way to achieve this type of a solution and optimize the system's investment is to build an Encryption Server to provide centralized encryption services for the entire database environment. This simplifies management and provides more control in a multi-application environment using many databases. This server can be optimized to perform cryptographic operations requested by the applications, giving the flexibility to allow applications to make multiple requests for cryptographic operations, while consolidating and implementing the cryptography in a consistent way. One great benefit of this solution is it offers one of the best secure key management strategies. This solution separates encryption keys from the encrypted data stored in the database (the encrypted data is injected into the database, but the keys never leave the Encryption Server) providing another layer of protection for the database. By contrast, Scenario One stores keys in the database with the encrypted data allowing an attacker easy access to both the keys and encrypted data. In Scenario Two outlined by the diagram above, the Encryption Server adds another layer of protection between the database and the attacker. The keys in the Encryption Server must be found before the hacker can decrypt data. The goal is to harden the Encryption Server against intrusion so that if anyone gained access to sensitive data in the database, they would find this text encrypted.

The Second Scenario is more secure because:

- Encryption keys are stored in hardware, separately from the encrypted text.
- End-to-end encryption is applied between the client and Encryption Server. Encrypted information is simply injected into the database.

The solution requires:

- Protecting the application and Encryption Server with an authentication strategy, preferably strong authentication, allows only authorized users to decrypt sensitive information by accessing keys stored in the Encryption Server.
- Hardening the Encryption Server against intrusion by monitoring it for suspicious activity and auditing event logs regularly [6].

4. MARS Encryption Algorithm

MARS supports 128-bit blocks and a variable key size. It is designed to take advantage of the powerful

operations supported in today's computers, resulting in a much improved security/performance tradeoff over existing ciphers. Specifically, in MARS we use a unique combination of S-box lookups, multiplications and data-dependent rotations. MARS has a heterogeneous structure, with cryptographic core rounds that are wrapped by simpler mixing rounds. The cryptographic core rounds provide strong resistance to all known cryptanalytical attacks, while the mixing rounds provide good avalanche and offer very wide security margins to thwart new attacks.

4.1 Description of MARS

MARS takes as input four 32-bit plaintext data words A, B, C, D and produces four 32-bit ciphertext data words A', B', C', D'. The cipher is word-oriented, in that all the internal operations are performed on 32-bit words. MARS is a type-3 Feistel network, divided into three phases: a 16-round "cryptographic core" phase wrapped with two layers of 8-round "forward" and "backwards mixing" in Figure 1. The cryptographic core rounds provide strong resistance to all known cryptanalytical attacks, while the mixing rounds provide good avalanche and offer very wide security margins to thwart new attacks.

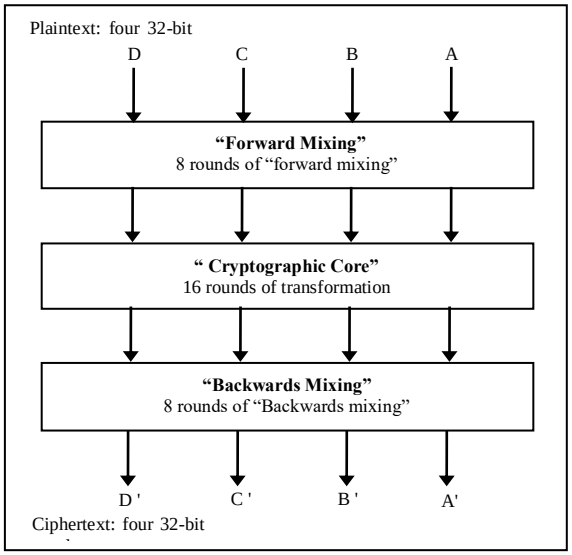

Figure 1: High-level structure of MARS encryption procedure

4.2 Key Setup

MARS accepts a variable size user-supplied key ranging from 4 to 14 words (i.e., 128 to 448 bits). MARS uses a key expansion procedure to "expand" the user-supplied key (consisting of n 32-bit words, where n is any number between 4 and 14) into a key array K[ ] of 40 words for the encryption/decryption operation.

4.3 MARS Key Expansion

The MARS key expansion procedure expands the user-supplied key ranging from 4 to 14 words into a 40-word key for use in the encryption/decryption operation. The key expansion procedure consists of three steps (in Figure 2). The first step is "linear expansion" which expands the original user-supplied key to forty 32-bit words using a simple linear transformation. The second step is "S-box based key stirring" which stirs the expanded key using seven rounds of a type-1 Feistel network to destroy linear relations in the key. Then a "multiplication key-word modifying" step examines the key words which are used in the MARS encryption/decryption operation for multiplication and modifies them if needed. The pseudo-code in Figure 3 shows the key expansion operation of MARS in detail. In the pseudo-code c∧d denotes bitwise-and of the two words c and d.

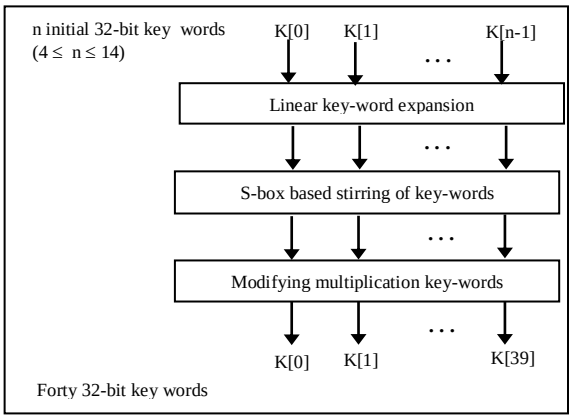

Figure 2: Key expansion procedure of MARS

// Initialize T[ ] With the Original Key Data k[ ]
T[0 … n-1] = k[0 … n-1], T[n] = n, T[n+1 … 14] = 0
For j = 0, 1, … ,3 do {
// Linear Key-Word Expansion
For i = 0, 1, … ,14 do { T[i] = T[i] ⊕ ((T[i-7 mod 15] ⊕ T[i-2 mod 15]) <<< 3) ⊕ (4i+j) **}**
// S-box Based Stirring of Key-Words
Repeat 4 times { For i = 0, 1, … ,14 do { T[i] = (T[i] + S[low 9 bits of T[i-1 mod 15]]) <<< 9 **} }**
// Store Next 10 Key-Words into K[ ]
For i = 0, 1, … ,9 do { K[10j+i] = T[4i mod 15] **}**
}
// Modifying Multiplication Key-Words
B[ ] = {0xa4a8d57b; 0x5b5d193b; 0xc8a8309b; 0x73f9a978}
For i = 5, 7, … ,35 do {
j = least two bits of K[i]
w = K[i] with both of the lowest two bits set to 1
Compute a word mask M as follows:
M = 0
M_n = 1 if w_n belongs to a sequence of 10 consecutive 0's or 1's in w, and also $2 \le n \le 30$ and $w_{n-1} = w_n = w_{n+1}$
r = least five bits of K[i-1]
p = B[j] <<< r
K[i] = w ⊕ (p ∧ M)
}
NOTE: x_i denotes the i-th bit in the 32-bit word x.

Figure 3: MARS key expansion pseudocode

4.4 MARS Cipher

The MARS cipher uses a variety of operations to provide a combination of high security, high speed, and implementation flexibility. Specifically, it combines exclusive-or (xor), addition, subtractions, multiplications, and both fixed and data-dependent rotations. MARS also uses a single (S-box) table of 512 32-bit words to provide good resistance against linear and differential attacks, as well as good avalanche of data and key bits. This S-box is also used by the key expansion procedure. Sometimes the S-box is viewed as two tables, each of 256 entries, denoted by S_0 and S_1. In the design of the S-box, we generated the entries in a "pseudo-random fashion" and tested that the resulting S-box has good differential and linear properties.

4.5 MARS Encryption

The pseudo-code in Figure 4 shows the encryption operation of MARS in detail. The operations used in the cipher are applied to 32-bit words, which are viewed as unsigned integers. This pseudo-code uses the following notations. Number the bits in each word from 0 to 31, where bit 0 is the least significant (or lowest) bit, and bit 31 is the most significant (or highest) bit. Denote by c⊗d a bitwise exclusive-or of the two words c and d. We denote by c+d addition modulo 2^{32}, by c-d subtraction modulo 2^{32}, and by c×d multiplication modulo 2^{32}. Also, c<<<d and c>>>d, denote cyclic rotations of the 32-bit word c by d positions to the left and right, respectively.

```
// Forward Mixing
(A,B,C,D) = (A,B,C,D)+(K[0],K[1],K[2],K[3])
For i=0 to7do {
        B = (B ⊗ S0[A]) + S1[A>>>8]
        C = C + S0[A>>>16]
        D = D ⊗ S1[A>>>24]
        A = (A>>>24) + B(if i=1,5) + D(if i=0,4)
        (A,B,C,D) = (B,C,D,A)
}

// Cryptographic Core
For i = 0 to 15 do {
        R = ((A<<<13) × K[2i+5]) <<< 10
        M = (A + K[2i+4]) <<< (low 5 bits of (R>>>5))
        L = (S[M] ⊗ (R>>>5) ⊗ R) <<< (low 5 bits of R)
        B = B +L(if i<8) ⊗ R(if i≥8)
        C = C + M
        D = D ⊗ R(if i<8) + L(if i≥8)
        (A,B,C,D) = (B,C,D,A<<<13)
}

// Backwards Mixing
For i = 0 to 7 do {
        A = A - B(if i=3,7) - D(if i=2,6)
        B = B ⊗ S1[A]
        C = C - S0[A<<<8]
        D = (D - S1[A<<<16]) ⊗ S0[A<<<24]
        (A,B,C,D) = (B,C,D,A<<<24)
}
(A,B,C,D) = (A,B,C,D) - (K[36],K[37],K[38],K[39])
```

Figure 4: MARS encryption pseudo code

NOTE: S0[X] and S1[X] use low 8 bits of X. S[X] uses low 9 bits of X.
S is the concatenation of S0 and S1.

4.6 MARS Decryption

The decryption operation of MARS is the inverse of the encryption operation and the code for decryption is similar to the code for encryption [7].

5. Implementation of the System

The proposed system, where in users are assigned to either of two levels, U_1 (low) and U_2 (high).The database objects are classified into public and private data. Users in U_1 have access right only to public data, whilst those in U_2 have access right to both public data and private data.

Public Data: are non-sensitive data that forms the bulk of the database and are open to all users for access.

Private Data: are sensitive data that has restricted access. Example treatment record and investigation results of patients are considered a sensitive data not to be disclosed to all users. But user such as Administrator who has need-to-know treatment of all patients should be granted access privilege to patients' treatment record and investigation result.

The system structure of the model is shown in Figure 5. Basically the model was divided into three layers: The first layer is the user interface layer, which contains two blocks, one for low level users (U_1) and the other for high level users (U_2). All users posses a unique key K in the form of a certificate that they use when accessing their encrypted private data in the database. The second layer is the database management layer, which also contains two main blocks, one that implements the mandatory access control (MAC) to the database, and another that houses a tamper-free controller that is closely linked with a trusted subject.

The functions of the controller (KC) are:

- To generate and encryption key K for encrypted private data.
- To encrypt private data before being storage in the database.
- To decrypt private data in the response to users queries that satisfied security requirements.
- To perform integrity check on users returned data.
- To facilitate authentication of users when necessary.

The trusted-subject (TS) is in charge of:

- Registering new users record
- Deleting users and data
- Declassifying users and data
- Updating classified and private data.

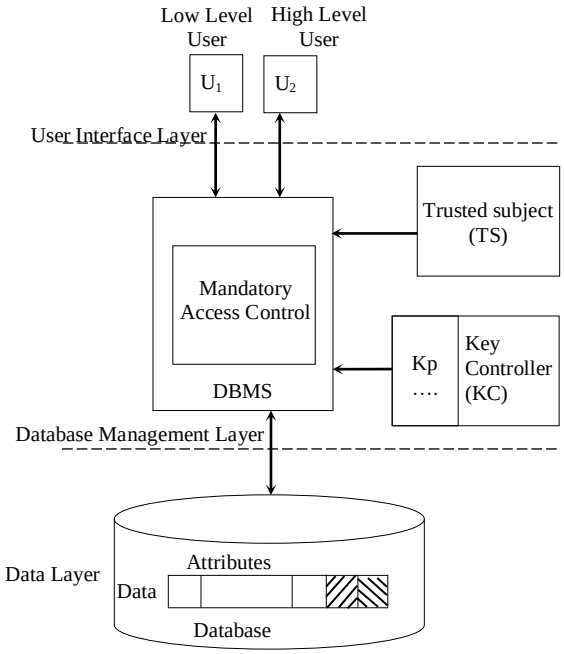

Figure 5: Structure of the Proposed System

The bottom layer contains the database. In order to facilitate fast retrieve of data, the database system stores classified data in the clear, whilst private data are stored in encrypted form. The first field of every record uniquely identifies the record, and serves as its primary key (ID). Primary key should not be confused with cryptographic keys used to decrypt ciphertext.

5.1 Database Security Viewpoint

It is described that how the system operates to provide security to the database. As mentioned earlier private data is stored in encrypted form in the database. If an intruder manages to circumvent the mandatory access control mechanism and penetrate the database, sensitive data are still concealed from them as they lack the necessary decryption keys. This provides confidentiality to sensitive data stored in the database. To control user's access to information in the database, the MAC maintains an authorized view of the database for all users.

To maintain the integrity of the database, users are constrained from carrying out direct access operations that may change the state of the database. Such operations include Create, Update, Delete, and change of Classification. Request for such operations should be forward to the trusted subject who evaluates them with respect to security violations and if safe allows their execution.

5.2 Key Management

The security of enciphered sensitive data in the database depends on the protection of the keys. For this reason, management of keys is vital to the overall security of the database system. The proposed scheme makes use of encryption keys K for private data key.

5.3 Key Generation

The method of key generation chose the key at random. A source of generating random key values is through pseudo-random key generators with different seed startup values. This forms the bases for generating the key K.

5.4 Key Distribution

The generated private data keys are stored in a tamper free controller. A copy of a private data key K is sent to the owner of the private data as a form of certificate. Request for a private data should be accompanied with its certificate for such request to be honored. Users can exchange their certificate with others who they wish to allow view to their private data, and can later request for an update. However only a private data owner can request update to private data or it certificate. Thus the controller must authenticate the originator for update to private data.

5.5 Encryption and Decryption

The private data elements are encrypted or decrypted using MARS Symmetric-key Encryption technique. Where data is less than the 16-byte block size of MARS, data is replicated as many times as necessary to fill the block. If data however exceeds the block size, then the encryption is performed using cipher block chaining with initialization block. Figure 5 and 6: illustrates the encryption and decryption of private data element respectively.

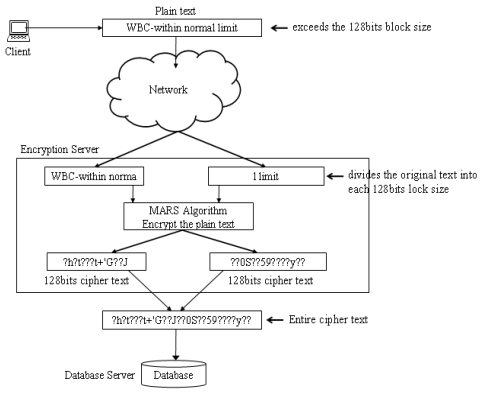

Figure 5: Encryption process of Private data

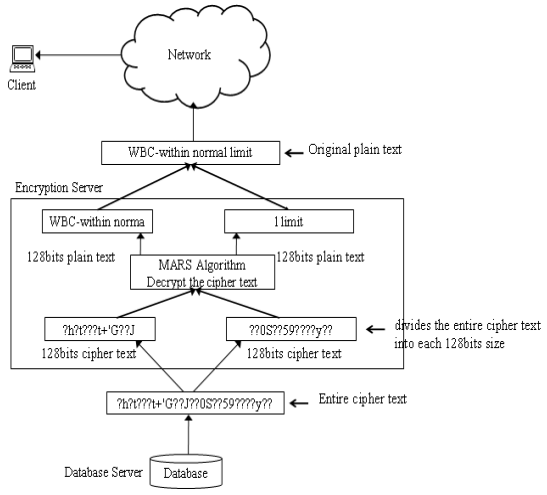

Figure 6: Decryption process of Private data

6. Conclusion

This paper investigates the role cryptograph can play in database security. The proposed database encryption scheme provides maximum security to the database, whilst the added time cost for encryption and decryption is very minimal. All aspects of security concerned from confidentiality, access control, integrity, authentication to non-repudiation were addressed.

To reduce the time spent on encryption and decryption, the scheme divides the database into non-sensitive (classified) and sensitive (private) data. Non-sensitive (classified) data, which forms the bulk of the database, are stored in the clear, facilitating their fast retrieval. Although private data are stored in encrypted form, their decryption process is very fast, as only one key is needed to decrypt a whole column (attribute) of encrypted classified data. Also, although accessed encrypted private data has to be decrypted separately using their unique keys. Requests for private data are very seldom and are carried out only once on a while. This makes the time cost for their encryption and decryption to have less significance on the overall performance of the scheme.

The limitation of the scheme is that queries such as sums, averages, counts and other statistical functions that aggregate over a range of data in the database cannot be performed directly. However users themselves at the user interface layer can do such task.

References

[1] Denning, D.E.,1983
"Field Encryption and Authentication."
Proc.of CRYPTO 8.9,Plenum Press.

[2] Denning, D. E., 1984.
"Cryptographic checksums for multilevel data security." Proc. of Symp.on Security and Privacy. IEEE Computer Society, pp: 52-61.

[3] Downs, D. and G.J. Popek, 1977.
"A Kernel Design for a Secure Data Base Management System."
Proc.3rd Conf. Very Large Data Bases. IEEE and ACM, New York, pp: 507-514.

[4] Davida, G.I, D.L. Wells and J.B. Kam, 1981.
"A Database Encryption System with sub keys."
ACM Trans. On Database Systems 6:2.

[5] Z.Yang, S.Sesay, J.Chen, D.Xu
"A Secure Database Encryption Scheme"
Department of Telecommunication and Information Technology. Huazhong University of Science and Technology, Wuhan, 430074, Peoples Republic of China

[6] Securing Data at Rest:
"Developing a Database Encryption Strategy"
A White Paper for Developers, e-Business Managers and IT

[7] "The MARS Encryption Algorithm"
C. Burwick [c] , D.Coppersmith [a] , E.D'Avignon [c] ,
R. Gennaro [a] , S.Halevi [a] , C.Jutla [a] , S.M. Matyas [c] ,
L. O'Connor [d] , M.Peyravian [b] , D.Safford [a] , N.Zunic [c]
[a] IBM T. J. Watson Research, Yorktown Heights, NY 10598, USA
[b] IBM Corporation, Research Triangle Park, NC 27709, USA
[c] IBM Corporation, Poughkeepsie, NY 12601, USA
[d] IBM Zurich Research, Rueschlikon, Switzerland
August 27, 1999